An Open-Source Time-Domain Wave-Based Room Acoustic software in Python Based on the Nodal Discontinuous Galerkin Method

Wang, Huiqing; Palha, Artur; Hornikx, Maarten

Publication date
2024

Document Version
Final published version

Citation (APA)
Wang, H., Palha, A., & Hornikx, M. (2024). *An Open-Source Time-Domain Wave-Based Room Acoustic software in Python Based on the Nodal Discontinuous Galerkin Method*. Paper presented at 53rd International Congress and Exposition on Noise Control Engineering, Nantes, France.

An Open-Source Time-Domain Wave-Based Room Acoustic software in Python based on the nodal discontinuous Galerkin method

Huiqing Wang[1]
Department of the Built Environment, Eindhoven University of Technology
Eindhoven, Netherlands

Artur Palha
Delft Institute of Applied Mathematics, Delft University of Technology
Delft, Netherlands
eScience Center
Amsterdam, Netherlands

Maarten Hornikx
Department of the Built Environment, Eindhoven University of Technology
Eindhoven, Netherlands

ABSTRACT

In this work, we introduce an open-source implementation of a time-domain wave-based room acoustic modeling software package, named `DG_RoomAcoustics`. *In this software, the linear acoustic equations are spatially discretized by the nodal discontinuous Galerkin method, and are integrated in time by either the explicit Runge-Kutta or the arbitrary high-order derivatives (ADER) integration schemes. Following the principles of object-oriented programming paradigm, the software is structured to ensure generic applicability and to facilitate future extensions with additional functionalities (e.g., different time integration schemes, boundary conditions). A comprehensive presentation of the physical and numerical aspects of the problem is provided. A detailed exposition of the code structure and components is presented, all of which are released under an open-source license, fostering community feedback and collaborative contributions for ongoing improvements. A brief overview of the current capabilities of the software is introduced. Future work regarding possible functionality extensions and performance optimizing are discussed.*

1. INTRODUCTION

The acoustic quality of indoor environments significantly influences human communication, comfort, and health. Consequently, accurate simulation of room acoustics plays a crucial role in the design and assessment of buildings, ensuring spaces meet the desired auditory requirements. However, the complexity of sound propagation demands sophisticated modeling tools that are both accurate and efficient to various acoustic scenarios. This need underscores the importance of developing advanced room acoustics simulation software, particularly tools that are open-source and readily accessible to the broader community of researchers, acoustic consultants, educators, and other stakeholders.

[1]h.wang6@tue.nl

Traditional room acoustic simulation tools [1,2] have largely relied on geometrical acoustics methods, which, while useful for certain applications, do not adequately capture complex wave phenomena such as diffraction, scattering, and mode effects that are crucial at lower frequencies. In contrast, wave-based methods [3–5] offer a more detailed and physically accurate modeling of sound fields, but their computational intensity and complexity have limited their adoption in practical applications. Hybrid approaches [6, 7] attempt to combine the strengths of both, yet there remains a substantial gap in tools that are both freely accessible and capable of providing high-fidelity simulations across the entire frequency range of interest.

Aiming to address these challenges, our work introduces an open-source, time-domain wave-based room acoustics modeling software, named `DG_RoomAcoustics`, which implements the nodal discontinuous Galerkin method for solving the linear acoustic equations in Python. This initiative not only seeks to replicate the validated functionalities of an existing in-house simulation tool developed in MATLAB, during the author's Ph.D. at Eindhoven University of Technology [8], but also serves as a platform to maximize the software's accessibility and to encourage wider community participation for further innovations and improvements in the field. The programming language of Python is chosen for this project, since its widespread popularity among scientists and engineers, combined with its comprehensive libraries and supportive community, makes it the ideal community-driven platform for developing and disseminating advanced acoustic modeling tools. Through this work, we aim not only to advance the state of the art in acoustic simulation techniques but also to cultivate a sustainable and collaborative ecosystem where users can contribute to and derive benefits from ongoing developments. This open-source project is envisioned to bridge the existing division between academic research and practical engineering applications, gathering the collective expertise of the community to foster the evolution of room acoustics modeling.

In the following sections, we introduce the theoretical foundation of the employed wave-based methods, outline the design and functionalities of this software, briefly discuss the steps to run a simulation scenario and conclude with future directions for this ongoing project.

2. THEORETICAL FOUNDATION

2.1. Linear Acoustic Equations

Under the assumption of lossless and constant propagation medium, the sound propagation in room acoustics is governed by the following linear acoustic equations:

$$\frac{\partial \boldsymbol{q}}{\partial t} + \nabla \cdot \boldsymbol{F}(\boldsymbol{q}) = \frac{\partial \boldsymbol{q}}{\partial t} + \boldsymbol{A_j}\frac{\partial \boldsymbol{q}}{\partial x_j} = \boldsymbol{0}, \tag{1}$$

where $\boldsymbol{q}(\boldsymbol{x}, t) = [u, v, w, p]^{\mathrm{T}}$ is the acoustic variable vector, containing the particle velocity component $[u, v, w]$ and the sound pressure p. The flux $\boldsymbol{F}$ is given as

$$\boldsymbol{F} = [\boldsymbol{A}_x\boldsymbol{q}, \boldsymbol{A}_y\boldsymbol{q}, \boldsymbol{A}_z\boldsymbol{q}], \tag{2}$$

where the constant flux Jacobian matrix $\boldsymbol{A_j}$ reads

$$\boldsymbol{A_j} = \begin{bmatrix} 0 & 0 & 0 & \frac{\delta_{xj}}{\rho} \\ 0 & 0 & 0 & \frac{\delta_{yj}}{\rho} \\ 0 & 0 & 0 & \frac{\delta_{zj}}{\rho} \\ \rho c^2\delta_{xj} & \rho c^2\delta_{yj} & \rho c^2\delta_{zj} & 0 \end{bmatrix}, \tag{3}$$

with coordinate index $j \in [x, y, z]$. δ_{ij} denotes the Kronecker delta function. ρ is the constant air density and c is the constant speed of sound.

2.2. Nodal Discontinuous Galerkin Method

To spatially discretize the linear acoustic equations Eq. (1), the nodal discontinuous Galerkin (DG) method [9] is used due to its several theoretical advantages. Primarily, the DG method allows for high-order accuracy and flexibility in handling geometrically complex domains, which are common in room acoustic applications. Unlike continuous finite element methods, the DG method treats discontinuities between elements explicitly, making it highly effective in capturing wave phenomena at interfaces.

Let Ω_h be a set of simplices, D^k, with $k = 1,\dots,K$, that are geometrically conformal and tessellate the domain Ω, providing a discretization of the computational domain, i.e., $\Omega_h := \bigcup_{k=1}^{K} D^k$. The local solution $\boldsymbol{q}_h^k(\boldsymbol{x},t)$ in element D^k, where subscript h denotes the numerical approximation, is given by:

$$\boldsymbol{q}_h^k(\boldsymbol{x},t) = \sum_{i=1}^{N_p} \boldsymbol{q}_h^k(\boldsymbol{x}_i^k,t)\, l_i^k(\boldsymbol{x}), \quad \boldsymbol{x} \in D^k, \tag{4}$$

where $\boldsymbol{q}_h^k(\boldsymbol{x}_i^k,t)$ are the unknown nodal values at the points $\boldsymbol{x}_i^k$, with $i = 1,\dots,N_p$, $l_i^k(\boldsymbol{x})$ is the multi-dimensional Lagrange polynomial basis of order N, which satisfies $l_i^k(\boldsymbol{x}_j^k) = \delta_{ij}$, and N_p is the number of local basis functions (or nodes) inside a single element and is equal to $(N+3)!/(N!3!)$ for simplex elements. After the Galerkin projection and integrating by parts twice, the semi-discrete nodal DG formulation of Eq. (1) reads:

$$\int_{D^k} \left(\frac{\partial \boldsymbol{q}_h^k}{\partial t} + \nabla \cdot \boldsymbol{F}_h^k(\boldsymbol{q}_h^k) \right) l_i^k \mathrm{d}\boldsymbol{x} = \int_{\partial D^k} \boldsymbol{n} \cdot \left(\boldsymbol{F}_h^k(\boldsymbol{q}_h^k) - \boldsymbol{F}^* \right) l_i^k \mathrm{d}\boldsymbol{x}, \tag{5}$$

where $\boldsymbol{n} = [n_x, n_y, n_z]$ is the outward normal vector of the element surface ∂D^k. $\boldsymbol{F}^*$ is the numerical flux across element intersection ∂D^k, and DG approaches are distinguished by the choice of numerical flux $\boldsymbol{F}^*$, in this study, the upwind numerical flux is used throughout the whole domain because of its low dispersive and dissipation error [4].

2.3. Time-Domain Impedance Boundary Condition Formulation

A critical aspect of accurate room acoustic modeling is the incorporation of realistic boundary conditions. In our software, the locally-reacting time-domain impedance boundary condition formulation is used [10]. This boundary formulation necessitates the input of complex-valued impedance values, denoted as $Z_s(\omega)$, across the relevant angular frequency range ω. Firstly, the plane wave reflection coefficient $R(\omega)$ at normal incidence angle is calculated using the impedance values as:

$$R(\omega) = \frac{Z_s(\omega) - 1}{Z_s(\omega) + 1}. \tag{6}$$

Then, the resulting plane-wave reflection coefficient $R(\omega)$ is fitted to a multi-pole model through the vector-fitting algorithm [11], enabling the simulation to accurately reflect the frequency-dependent behavior of sound absorption by different materials. The multi-pole model can be directly transformed to the time-domain. The coupling of this time-domain impedance boundary condition with the discontinuous Galerkin discretization is achieved through the characteristic waves of the upwind flux along the boundary, where the reflected characteristic wave is expressed as the convolution between the reflection coefficient at normal incidence and the incident characteristic wave. The convolution is calculated by integrating a series of first-order auxiliary differential equations in time. More detailed discussions are described in Ref. [10].

2.4. Time Integration Schemes

To integrate the acoustic equations in time, our software incorporates two time integration schemes: explicit 2N-storage Runge-Kutta [12] and ADER (Arbitrary high-order DERivative) [13].

Extension to any other explicit Runge-Kutta method is "trivial". The explicit Runge-Kutta scheme is favored for its simplicity and robustness, offering a straightforward approach to advancing the solution in time. On the other hand, the ADER scheme provides an alternative capable of of matching the approximating order of the spatial discretization (arbitrary order), therefore enabling the construction of a unified framework for arbitrary high order approximation in both time and space. The choice between these schemes is made based on the specific requirements of the simulation, balancing between computational efficiency and the desired level of detail in the simulation outcomes. The simulations can be initiated with any initial condition that can be expressed with the discretization. For ease of use, we have implemented an initialization of Gaussian-shaped pressure conditions:

$$p(\boldsymbol{x}, t=0) = \mathrm{e}^{\frac{-\ln 2}{b^2}(\boldsymbol{x}-\boldsymbol{x_s})^2}, \tag{7a}$$

$$\boldsymbol{v}(\boldsymbol{x}, t=0) = \boldsymbol{0}, \tag{7b}$$

with $\boldsymbol{x_s}$ the source coordinates and b the half-bandwidth of this Gaussian pulse. It represents a source of monopole type. The value of b controls the highest-achievable frequency range of the source. A smaller b indicates a source spectrum up to a higher frequency, which necessitates a finer mesh.

3. SOFTWARE DESIGN AND IMPLEMENTATION

This section elucidates the architecture and the implementation particulars of `DG_RoomAcoustics`. The source code is hosted on a public Github repository [2]. The software is designed with a modular approach, exploiting the object-oriented programming features to provide a comprehensive and extensible framework. Each of core classes encapsulate distinct functionalities, and their attributes and methods are carefully documented with Sphinx [14] to ensure clarity of purposes and ease of use. The following subsections expound on the implementation details of each of the core classes, their specific roles and interactions within the software, as shown in the unified modeling language (UML) class diagram of Fig. 1.

3.1. Mesh Class

The `Mesh` class, as implemented in the sub-module mesh.py, provides functionality to process and access mesh data generated by different mesh generators. It uses an external tool `meshio` [15] to import and manage the mesh file (mesh_file) provided by the user. Moreover, user needs to provide the necessary tagging information of various boundary surfaces corresponding to specific boundary conditions, which is stored in the variable BC_labels. The created mesh object contains all the necessary information of the spatial domain for simulation.

3.2. BoundaryCondition Class

The `BoundaryCondition` class, as implemented in the sub-module boundary_condition.py, is an abstract class that outlines the necessary structure and functionalities for implementing various boundary conditions. Tailored to the room acoustic modelling applications, an `AbsorbBC` class inherits the `BoundaryCondition` class to simulate common absorptive boundary conditions in room acoustics. The fitting parameters of the multi-pole model, as described in Section 3, are supposed to be provided by the user through the variable BC_para following a specific format. By construction an abstract boundary condition class with a general enough interface, we provide the possibility of expandability and customisation. Any user can develop their own boundary condition implementation and, as long as they fit the provided interface, the solver will be able to directly use it without any additional changes.

[2] https://github.com/Building-acoustics-TU-Eindhoven/edg-acoustics

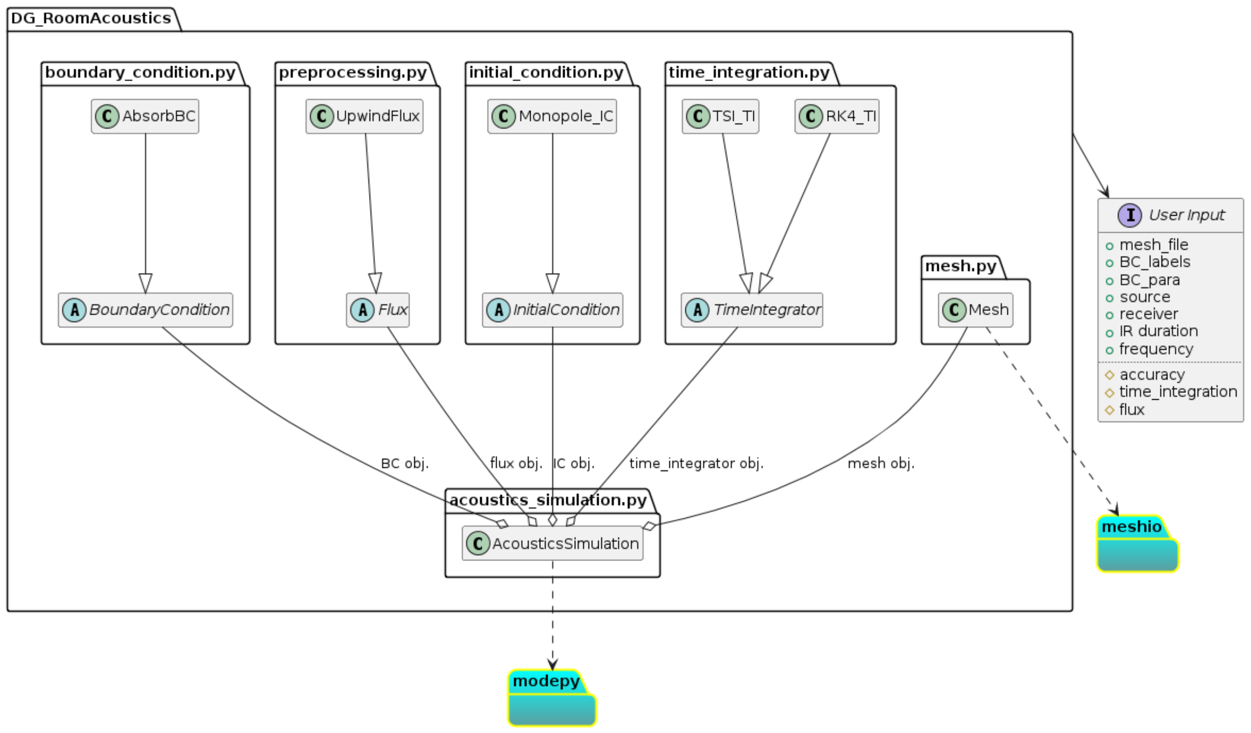

Figure 1: Module and class diagram for `DG_RoomAcoustics`

3.3. InitialCondition Class

The `InitialCondition` class, as implemented in the sub-module initial_condition.py, is an abstract class that outlines the necessary structure and functionalities for implementing various initial conditions. It establishes the initial acoustic state within the domain, as the simulation's starting point. At this moment, a `Monopole_IC` class is implemented to simulate a monopole source. The users (and contributors) can implement any other initial conditions and they will effortless work with the existing code.

3.4. Flux Class

The `Flux` class, as implemented in the sub-module preprocessing.py, is an abstract class that outlines the necessary structure and functionalities for implementing various numerical fluxes of DG. As mentioned in Section 2, the upwind flux is implemented inside the `UpwindFlux` class for its superior properties. It should be noted that the creation of flux object needs the normal vectors of all tetrahedra mesh elements as arguments, which are calculated by a method inside the `AcousticsSimulation` class.

3.5. TimeIntegrator Class

The `TimeIntegrator` class, as implemented in the sub-module time_integration.py, is an abstract class that outlines the necessary structure and functionalities for implementing various time integration schemes. The design of this abstract class is such that any time integrator can be implemented and will directly work with the solver. It has two child classes that exemplify the specific implementations of Taylor-series time integration (`TSI_TI` class) and fourth-order Runge-Kutta integration (`RK4_TI` class).

3.6. AcousticsSimulation Class

The AcousticsSimulation class, as implemented in the sub-module acoustics_simulation.py, serves as the fulcrum for running the acoustics simulation, accompanying the entire simulation procedure from initiation to execution. After taking the user input of acoustic medium property (density ρ and speed of sound c), a polynomial degree of the approximating finite element space and the created mesh object, it calls an external library modepy [16] to initialize the local system and to compute attributes such as the differentiation matrices and nodes distribution. Holding an aggregation relation to other classes, the AcousticsSimulation class has collection of mesh, BC, flux, IC and time_integrator objects as its attributes as shown in Fig. 1. The temporal evolution of the acoustic variables over specified time steps or total time is defined as an instance method within this class.

4. SIMULATION EXAMPLES AND SOFTWARE DOCUMENTATION

The software package DG_RoomAcoustics is designed to be user-friendly and accessible to a broad audience, including both end users and developers. The repository includes a detailed README file that provides a comprehensive guide on how to install and run the software. The software is accompanied also example scripts to guide users through the process of setting up and running room acoustic simulations. The documentation of implementation details is hosted on a dedicated website [3]. The documentation is continuously updated to reflect the latest changes and additions to the software, ensuring that users have access to the most up-to-date information.

To execute a simulation, the following steps should be followed:

1. Import the essential modules, e.g., Numpy and DG_RoomAcoustics.
2. Provide the boundary condition information in the correct format, represented as dictionaries BC_para and BC_labels.
3. Initialize the mesh object.
4. Set the source for the simulation with the InitialCondition class, indicating the source's position and characteristics.
5. Determine the degrees of polynomial approximation and the receiver locations as a Numpy array.
6. Create an AcousticsSimulation object with the defined parameters, including the medium's properties and the mesh object..
7. Configure the flux computation through the UpwindFlux class and establish the absorptive boundary conditions using the AbsorbBC class.
8. Initialize the time integrator object of the TimeIntegrator class.
9. Commence the simulation by invoking the time_integration method of the created AcousticsSimulation instance object for the total simulation duration.
10. Save the simulated impulse response in the desired format for further analysis and visualization.

[3] dg-roomacoustics.readthedocs.io

5. CONCLUSION AND FUTURE DIRECTIONS

This work has presented `DG_RoomAcoustics`, an open-source room acoustic modeling software package employing the nodal discontinuous Galerkin method for solving linear acoustic equations in Python. Designed with extensibility and community involvement in mind, `DG_RoomAcoustics` showcases the significant potential for advancing the state-of-the-art in acoustic simulations through collaborative development. Looking ahead, we anticipate several issues and directions for the software's growth and refinement:

– **Scientific Computing Ecosystem Compatibility:** Efforts will focus on ensuring `DG_RoomAcoustics`'s compatibility with the broad ecosystem of scientific computing in Python. The computational performance of the underlying matrix-vector and elementwise multiplication are planned to be optimized with advanced high-performance parallel computing frameworks.

– **Validation and Performance Analysis:** Comprehensive validation and performance analysis of the Python implementation are paramount. Future versions will include a detailed comparison with analytical solutions and experimental data to ensure the reliability and accuracy of the simulations.

– **User Interface Development:** Development of a user-friendly interface is underway, which will streamline the setup and execution of simulation scenarios, thereby making the tool more accessible to practitioners and researchers alike.

– **Advanced Features and User Feedback Adaptation:** Continuous integration of advanced features will be based on user feedback. Planned enhancements include the addition of complex boundary conditions, post-processing functionalities for room acoustic parameter calculations, diverse data export options, and result visualization via field (cut) views.

– **Automated Testing and CI/CD:** Implementing automated testing and Continuous Integration/Continuous Deployment (CI/CD) pipelines using GitHub Actions will enhance the software development lifecycle, ensuring robustness and maintainability.

Last but not least, the trajectory of `DG_RoomAcoustics` is aimed at not just software development but fostering a community where innovation is communal and contributions lead to shared success. We invite the research and development community to engage with `DG_RoomAcoustics`, to propel it as a tool of choice for room acoustic simulation.

ACKNOWLEDGEMENTS

This work is funded by the Dutch Research Council (NWO) under grant No. 19430. The authors would also like to acknowledge the support of eScience center of Netherlands under the grant of Small-Scale Initiatives in Software Performance Optimization (OpenSSI 2021b) .

REFERENCES

1. Odeon Web page. https://odeon.dk/, January 2024.
2. CATT-Acoustic Web page. https://www.catt.se/, January 2024.
3. Stefan Bilbao, Brian Hamilton, Jonathan Botts, and Lauri Savioja. Finite volume time domain room acoustics simulation under general impedance boundary conditions. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(1):161–173, 2016.
4. Huiqing Wang, Indra Sihar, Raúl Pagán Muñoz, and Maarten Hornikx. Room acoustics modelling in the time-domain with the nodal discontinuous Galerkin method. *The Journal of the Acoustical Society of America*, 145(4):2650–2663, 2019.

5. Finnur Pind, Allan P Engsig-Karup, Cheol-Ho Jeong, Jan S Hesthaven, Mikael S Mejling, and Jakob Strømann-Andersen. Time domain room acoustic simulations using the spectral element method. *The Journal of the Acoustical Society of America*, 145(6):3299–3310, 2019.
6. Alex Southern, Samuel Siltanen, Damian T. Murphy, and Lauri Savioja. Room Impulse Response Synthesis and Validation Using a Hybrid Acoustic Model. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(9):1940–1952, sep 2013.
7. Gudrun Oskarsdottir, Finnur Pind, and Jesper Pedersen. Industrial applications of hybrid room acoustic simulations using a combination of discontinuous galerking method and geometrical acoustic methods-benchmark case study. In *Audio Engineering Society Conference: AES 2024 International Acoustics & Sound Reinforcement Conference*. Audio Engineering Society, 2024.
8. Huiqing Wang. *Room acoustic modeling with the time-domain discontinuous Galerkin method*. PhD thesis, Built Environment, October 2021.
9. Jan S Hesthaven and Tim Warburton. *Nodal discontinuous Galerkin methods: Algorithms, Analysis and Applications*. Springer-Verlag, New York, 2007.
10. Huiqing Wang and Maarten Hornikx. Time-domain impedance boundary condition modeling with the discontinuous Galerkin method for room acoustics simulations. *The Journal of the Acoustical Society of America*, 147(4):2534–2546, 2020.
11. Bjorn Gustavsen and Adam Semlyen. Rational approximation of frequency domain responses by vector fitting. *IEEE Transactions on power delivery*, 14(3):1052–1061, 1999.
12. Mark H. Carpenter and Christopher A. Kennedy. Fourth-order 2N-storage Runge-Kutta schemes. In *NASA-TM-109112*, 1994.
13. Huiqing Wang, Matthias Cosnefroy, and Maarten Hornikx. An arbitrary high-order discontinuous Galerkin method with local time-stepping for linear acoustic wave propagation. *The Journal of the Acoustical Society of America*, 149(1):569–580, 2021.
14. Georg Brandl. Sphinx documentation. *URL http://sphinx-doc. org/sphinx. pdf*, 2021.
15. Nico Schlömer. meshio: Tools for mesh files.
16. Andreas Klöckner. modepy. https://github.com/inducer/modepy, 2024.